TELECOMMUNICATION TERMINAL COMPRISING TWO EXECUTION SPACES

The invention concerns the execution of programmes and applications on a computing device with user interface (keyboard, monitor, sound card, touch-sensitive area, mouse etc..) e.g. on a domestic gateway, a sales machine (public machine for example) or a telecommunications terminal (from PC to mobile phone).

Different approaches are known for implementing applications in telecommunications terminals.

For example the MIDP 2.0 profile on a virtual machine uses a security policy based on open standards, easy to use, placing no demand on the user, which takes into account the needs of each person concerned from development to execution (the concepts of user, operator, OEM, trusted third party are separated).

It affords protection of integrity and verification of the origin of applications during their downloading and execution, control of access to critical resources in accordance with a security policy, user alerting of operations in progress and can even request user opinion.

The security policy is taken into consideration in fairly simple manner at an API to be protected by having recourse to the "check permission" method of the "Midlet" class (figure 2).

This requires that the call function of the MIDP file should not be directly accessible from MIDP programmes (protected function).

The MIDP 2.0 security policy is well adapted to the needs of the various persons concerned. The possibility of requesting user opinion in accordance with a certain number of criteria (always, once, for one session, never) is highly advantageous.

Nonetheless its implementation raises two types of problems.

Firstly execution of the protected procedure is made in the same execution space as the caller programme, which facilitates the risk of "leaks". Let us imagine a ciphering service called by two midlets simultaneously, there is nothing to guarantee, if little attention is paid thereto, that one midlet may recover the content of the private key used by the other midlet.

The first problem is therefore lack of security, in particular for risk applications such as payment, signature or DRM applications for example.

Moreover, some feats have shown that with an implementation error it is possible to override this permission system.

The second problem of the MIDP profile is raised by the specifications of the MIDP profile itself. It is not adapted to the formal proof method for programmes. This raises a problem in some sectors (banking sectors in particular) in which a midlet cannot be modelled by formal methods, and hence cannot be checked by these methods.

In other words, no technique exists with which it is possible to prove validity, via formal methods, with respect to the specifications of a programme programmed in this profile.

Another profile, the STIP profile, is more especially adapted for providing access to security-orientated APIs, such as SIM access.

STIP virtual machines (figure 3) are used for the functioning of programmes especially written for the STIP profile.

The other asset of STIP is that its programming model and its APIs lend themselves well to analysis by formal methods.

This is what prompted its adoption by the banking sector since code conformity to specifications can be proved by formal method.

Therefore the STIP profile used in the banking sector, through its limitations, is adapted to programme proof checking.

However the STIP profile was arranged for closed systems (untrusted applications cannot be downloaded with impunity).

Therefore no security model is set up (in version 2.1.1. of the specification) and hence any STIP application (stiplet) can access any API of STIP type that has already been implemented.

The STIP profile is thus not adapted for producing terminals in which a user is likely to download and implement current applications such as games or various utility applications.

The purpose of the invention is to propose a configuration with which it is possible, in a telecommunications terminal, to implement various user applications and also applications requiring a high level of security.

The invention also sets out to facilitate the programming and implementing of applications, in particular by facilitating the certification of the proper functioning of newly programmed applications.

It is true that the principle is known of mobile telephones hosting two virtual machines in the physical form of two processors, one formed by the terminal itself and the other formed by the SIM card.

The SIM card checks high security requirements, whilst the processor of the terminal itself and its content are accessible to the user.

However, said implementation still has some major disadvantages.

Therefore another purpose of the invention is to propose a device which may or may not be network associated, in which advantage is taken of a securitized space and a non-securitized space, e.g. by allowing the securitized space to access user interfaces such as keyboard or monitor instead of the non-securitized space, and conversely allowing a non-securitized space to access a securitized communication with a known operator to guarantee said security. As security operator, particular mention may be made of telephony operators, mobile telephone operators in particular, banks, suppliers of multimedia items with selective or paying distribution, service providers requiring electronic signature via said device.

Suppliers of multimedia items with selective distribution are particularly "DRMs" 'Digital Rights Management) these servers delivering a content that typically relates to music, video, or games under licence, and in a file form which can be read under various constraints e.g. a certain number of times.

One purpose of the invention is to propose said means in which, in addition, there is certainty that the two associated executions spaces (one with a higher security level than the other) are effectively those which are intended or permitted to be associated with each other *ab initio*.

These purposes are achieved through the invention by means of a computing device with user interface comprising means for implementing a series of applications, these means including in particular a virtual machine/functioning profile execution space, the device comprising a second virtual machine/functioning profile execution space differing from the first by at least its virtual machine or its functioning profile, each execution space hosting applications, the

applications of the second execution space being applications with a specifically higher security level than the applications of the first execution space on account of the fact that the applications of the first execution space are applications set up and activated by the terminal user, whilst the applications of the second execution space are applications which cannot be modified by the terminal user, characterized in that the two execution spaces are hosted by a physical processing means which is arranged so that it cannot be separated into two parts without destroying this physical processing means.

The invention also proposes a method for implementing applications within a computing device with user interface, the method having recourse to means for implementing a series of applications, these means including in particular a virtual machine/functioning profile execution space and a second virtual machine/functioning profile execution space differing from the first by at least its virtual machine or its functioning profile, each execution space hosting applications, the applications of the second execution space being applications with a specifically higher security level than the applications of the first execution space on account of the fact that the applications of the first execution space are applications set up and activated by the terminal user, whilst the applications of the second execution space are applications which cannot be modified by the terminal user, characterized in that the two execution spaces are hosted by a physical processing means which is arranged so that it cannot be separated into two parts without destroying this physical processing means.

Other characteristics, purposes and advantages of the invention will become apparent on reading the following

detailed description which refers to the appended figures in which:

- figure 1 is a schematic illustrating a MIDP implementation of the prior art
- figure 2 is a schematic illustrating the implementation of protection means in a said MIDP implementation,
- figure 3 is a schematic illustrating a prior art STIP implementation
- figure 4 illustrates a functional configuration of an inventive terminal according to a preferred variant.

The particular embodiment described below makes it possible to take best advantage of both techniques, MIDP and STIP, given as an example within a coherent execution environment.

The "user" profile is formed therein, the MIDP profile. This profile is very popular in the world of cell phones for creating games and various utility applications. The user is able to download and execute applications found on the network in the same way as with a usual MIDP telephone. The MIDP profile therefore includes applications set up and activated by the users themselves.

Here the STIP profile forms an additional profile, and more specifically an "operator" profile. The STIP profile is well adapted to applications requiring a high security level, such as banking applications. Banking consortiums have already placed their trust in the possibility using formal methods to certify STIP applications for their implementation in electronic payment terminals (EPT).

With the present invention it is therefore possible to provide developers with an operator API assembly whose execution is ensured in an execution space appropriate for easy programming by these developers, a space of whether or not of same profile, and fully separate.

This embodiment therefore enables operators to provide a batch of securitized applications such as payments, signature or DRM, that is fully independent from the execution profile for "routine" applications.

The terminal shown figure 4 includes and causes to operate in harmony two virtual machines 100 and 200 of separate profiles P1 and P2 (or not separate). One 100 of the two machines is dedicated to user applications, the other 200 to operator applications.

The corresponding profiles P1 and P2, here respectively the MIDP profile and the STIP profile, are themselves respectively dedicated to user applications and operator applications.

Figure 4 shows two virtual machines 100 and 200.

The "user" virtual machine 100 can be used by the user to download, install, uninstall, execute, stop applications in the MIDP profile as and when desired. The applications 110 operating therein use the API 120 of this profile and an API "stub" having the same profile as the machine 100, this API stub being referenced 130 in figure 4.

The second machine, referenced 200 is the "operator" virtual machine: only the operator e.g. the mobile telephone operator or the internet operator (access supplier), via an over the Air (OTA) mechanism, is able to administrate this execution space.

The operator can, at will, install, uninstall, activate, deactivate therein applications 210 written as per the formalism of profile 100. These applications 210 have access to APIs 220 of profile P2 and to one or more high level APIs illustrated under reference 230 in figure 4.

These high level APIs 230 allow access to services offered by the profile of machine 100. API access, whether from the profile of machine 200 or the stub 230 to the profile

of machine 100 is made in accordance with the security model inherent in the profile of machine 200.

The API "stub" 130 is a high level API, expressed as per the programming model of profile 100, providing access to services offered by profile P2. API access, whether to the profile of the machine 100 or a stub 130 is made in accordance with the security model inherent in the profile P1 of machine 100.

The functioning of stubs 130 and 230 is as follows:

The call to an API of stub 130, 230 is converted into a flow of octets (called either serialization process or marshalling /unmarshalling).

This flow is received by a manager 140, 240 of the opposite profile via a communication channel 300, deserialized and converted into execution of a procedure in the remote profile. The return execution of this procedure is again serialized in the remote profile and passes again in the communication channel 300 between the two profiles P1 and P2 of machines 100 and 200, the reply is deserialized in the original profile and converted into a return call of the API "stub".

In this way there are two independent execution spaces each consisting here of a different machine and a different profile, and in very close relationship via the API stub 130 and 230.

As a variant, the two profiles P1 and P2 may be of same type e.g. two MIDP profiles or two STIP profiles for two different machines.

It will also be noted that it is possible to adopt two different profiles P1 and P2 within one same virtual machine.

This embodiment therefore offers a payment API to developers of MIDP applications, in which the payment itself

is made under the execution of a virtual STIP machine controlled by the operator.

In other words, a MIDP application, easily developed, could offer the user a means of payment by causing a payment application of machine 200 to operate via the communication channel 300. A MIDP application, through the invention, is therefore able to offer a payment functionality that is highly reliable.

The two execution spaces 100 and 200 each formed of a virtual machine/execution profile pair, differing from one another through the profile or the virtual machine, are both implemented however by one same physical processing device 400 (same hardware entity 400).

This processing device hosting the two execution spaces is unique in that it cannot be divided into two without destroying its functioning.

It is therefore impossible to physically separate the two execution spaces, and it is hence also impossible to associate a space thus separated with another space which is not authorized.

Said achievement with a single means is obtained for example by implementing the two execution spaces on one same integrated circuit forming a single processor.

It is thereby ensured that two environments, one securitized and the other non-securitized, are inseparable.

The security offered by an operator (telephony, banking, signature administration, multimedia distributor) is thereby improved whether to prevent security overriding in payment functions, to ensure confidentiality or non-usurpation of secret codes, to ensure reliability of electronic signatures or to monitor limited user rights for payable services.

Advantageously the P1, P2 profiles of each of the two execution spaces 100, P1, 200, P2 are respectively a STIP

profile and a profile forming part of the group made up of STIP, MIDP, OSGI and ".net" profiles.